EN. 601.467/667 Introduction to Human Language Technology Deep Learning I

Shinji Watanabe

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Today’s agenda

- Introduction of deep neural network
- Basics of neural network

Short bio

- Research interests
  - Automatic speech recognition (ASR), speech enhancement, application of machine learning to speech processing
- Around 20 years of ASR experience since 2001

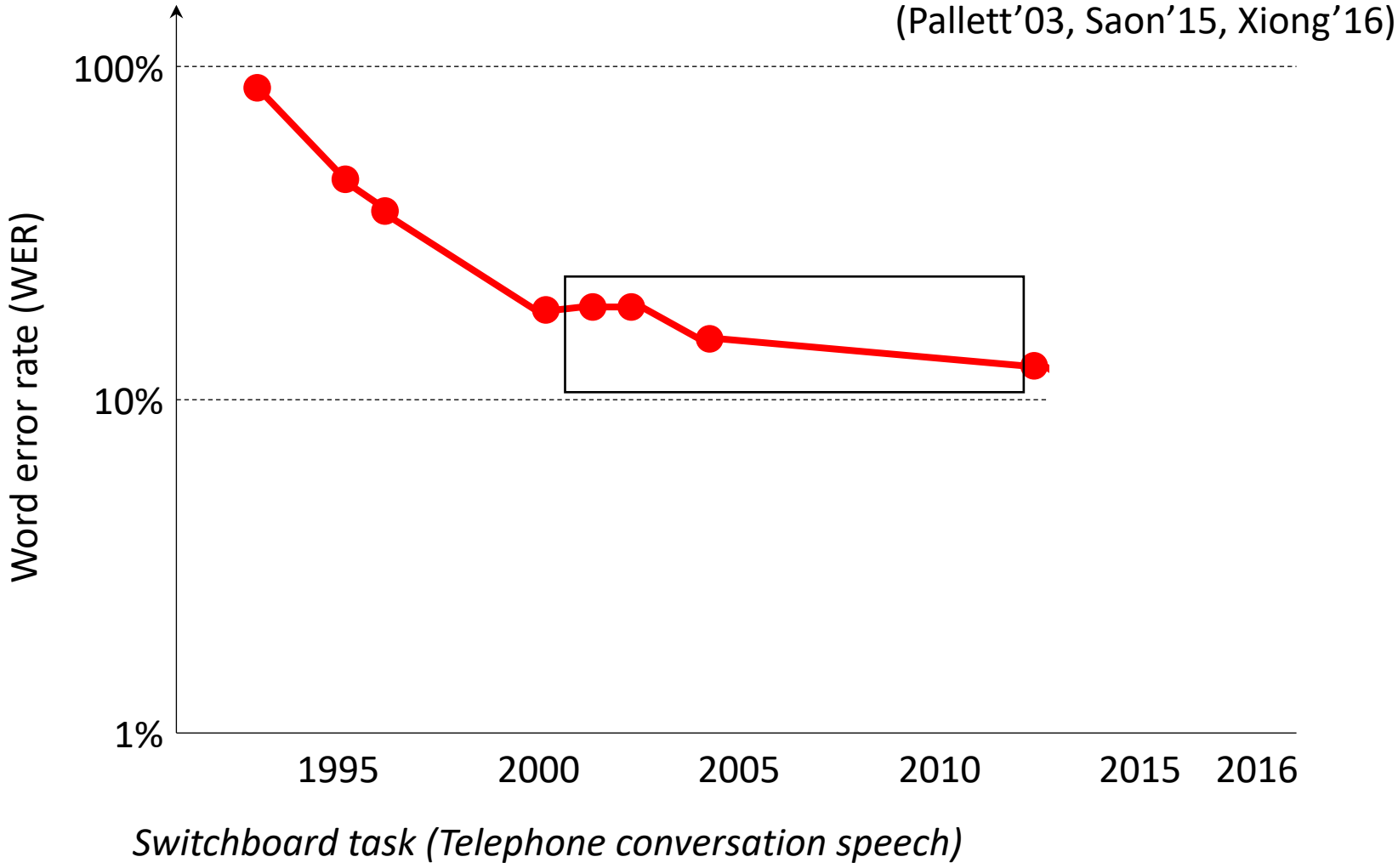
Speech recognition evaluation metric

- Word error rate (WER)
  - Using edit distance word-by-word:

```
Reference)
I want to go to the Johns Hopkins campus
Recognition result)
I want to go to the 10 top kids campus
```

  - # insertion errors = 1, # substitution errors = 2, # of deletion errors = 0 ➡ Edit distance = 3
  - Word error rate (%): Edit distance (=3) / # reference words (=9) * 100 = 33.3%
  - How to compute WERs for languages that do not have word boundaries?
    - Chunking or using character error rate

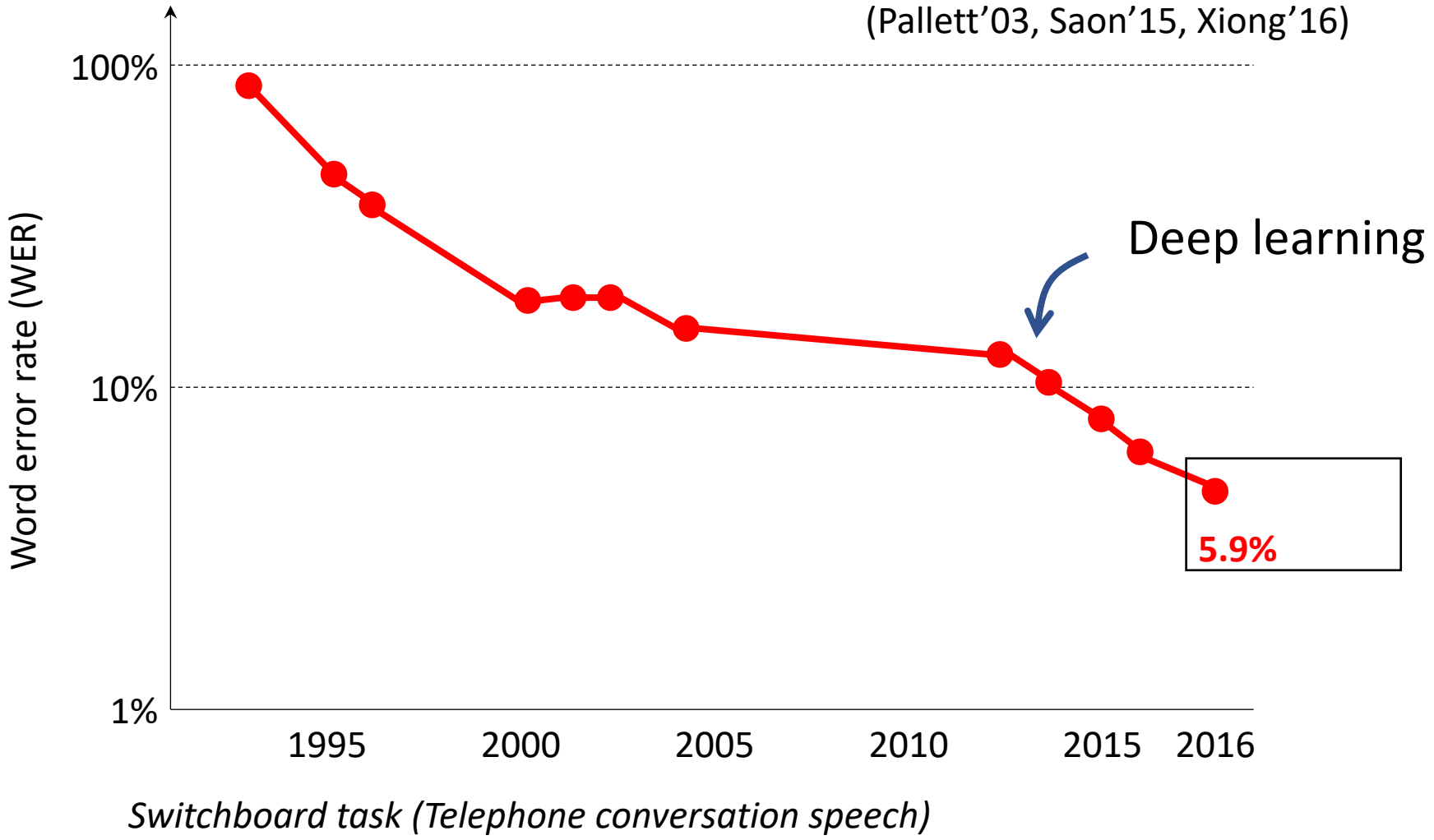
2001: when I started speech recognition….



Really bad age….

- No application
- No breakthrough technologies
- Everyone outside speech research criticized it…
- General people don’t know “what is speech recognition”

Now we are at


(Pallett'03, Saon'15, Xiong'16)
100%
10%
1%
Word error rate (WER)
Deep learning
5.9%
1995
2000
2005
2010
2015
2016
Switchboard task (Telephone conversation speech)

Everything was changed

- No application
- No breakthrough technologies
- Everyone outside speech research criticized it…
- General people don’t know “what is speech recognition”

Everything was changed

- ~~No application~~ **voice search, smart speakers**
- No breakthrough technologies
- Everyone outside speech research criticized it…
- General people don’t know “what is speech recognition”

Everything was changed

- ~~No application~~ **voice search, smart speakers**
- ~~No breakthrough technologies~~ **deep neural network**
- Everyone outside speech research criticized it…
- General people don’t know “what is speech recognition”

Everything was changed

- ~~No application~~ **voice search, smart speakers**
- ~~No breakthrough technologies~~ **deep neural network**
- ~~Everyone outside speech research criticized it…~~ **many people outside speech research know/respect it**
- General people don’t know “what is speech recognition”

Everything was changed

- ~~No application~~ **voice search, smart speakers**
- ~~No breakthrough technologies~~ **deep neural network**
- ~~Everyone outside speech research criticized it…~~ **many people outside speech research know/respect it**
- ~~General people don't know "what is speech recognition"~~ **now my wife knows what I'm doing**

Acoustic model

- From Bayes decision theory to acoustic model

$$\begin{aligned}\underset{W}{\operatorname{argmax}}\, p(W|O) &= \underset{W}{\operatorname{argmax}} \sum_{L} p(W, L|O) \\ &= \underset{W}{\operatorname{argmax}} \sum_{L} p(O|L, W)p(L, W) \\ &= \underset{W}{\operatorname{argmax}} \sum_{L} p(O|L)p(L|W)p(W)\end{aligned}$$

Acoustic model **Language model**

GMM/HMM

- Given HMM state j, we can represent the likelihood function as

$$p(\mathbf{o}_{t+1}|s_{t+1}=j,\Theta)=\sum_{k=1}^{K}p(\mathbf{o}_{t+1},v_{t+1}=k|s_{t+1}=j,\Theta)$$
$$=\sum_{k=1}^{K}\omega_{jk}\mathcal{N}(\mathbf{o}_{t+1}|\boldsymbol{\mu}_{jk},\mathbf{r}_{jk})$$

- **Deep neural network** acoustic model only replaces this GMM representation with a neural network

$$p(\mathbf{o}_{t+1}|s_{t+1}=j,\Theta)=\mathrm{DNN}(\mathbf{o}_{t+1}|s_{t+1}=j)$$

Problem

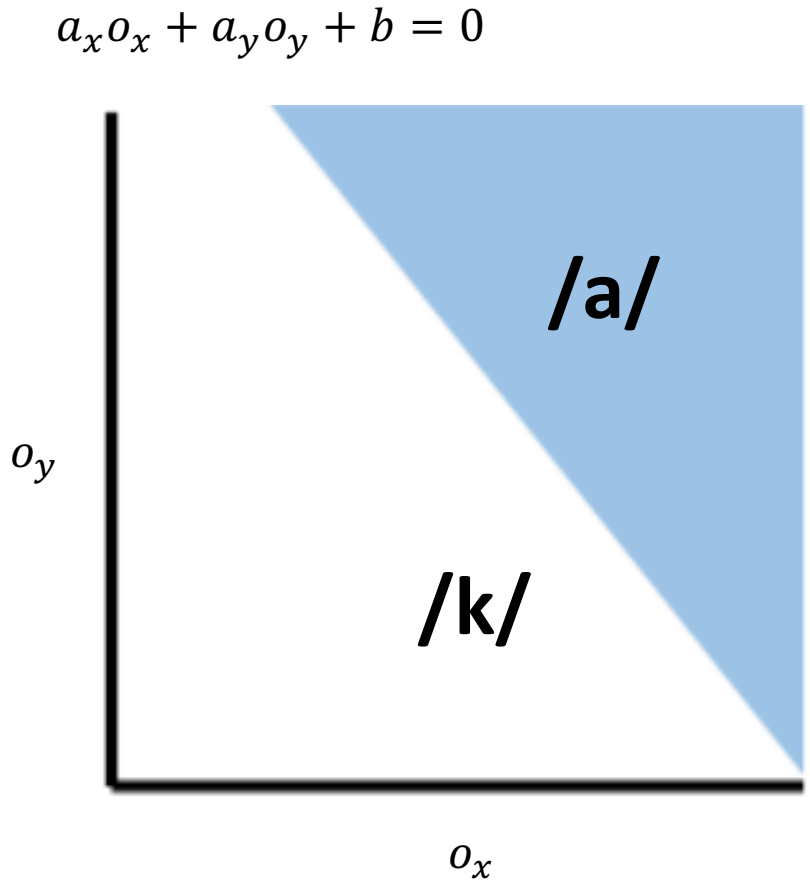
- Input MFCC vector: $\mathbf{o}_t$
- Output phoneme (or HMM state): $s_t = \{/a/, /k/, \ldots, \}$

- How to find a probabilistic distribution of $p(s_t|\mathbf{o}_t)$???
- We use a large amounts of pair data $\{\mathbf{o}_t, s_t\}_{t=1}^{T}$ to train the model parameter of the distribution

Very easy case

- We can use a linear classifier
  - /a/: $a_x o_x + a_y o_y + b \geq 0$
  - /k/: $a_x o_x + a_y o_y + b < 0$
- We can also make a probability with the sigmoid function $\sigma()$
  - $p(/\text{a}/|\mathbf{o}) = \sigma(a_x o_x + a_y o_y + b)$
  - $p(/\text{k}/|\mathbf{o}) = 1 - \sigma(a_x o_x + a_y o_y + b)$
- Sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



from http://cs.jhu.edu/~kevinduh/a/deep2014/140114-ResearchSeminar.pdf

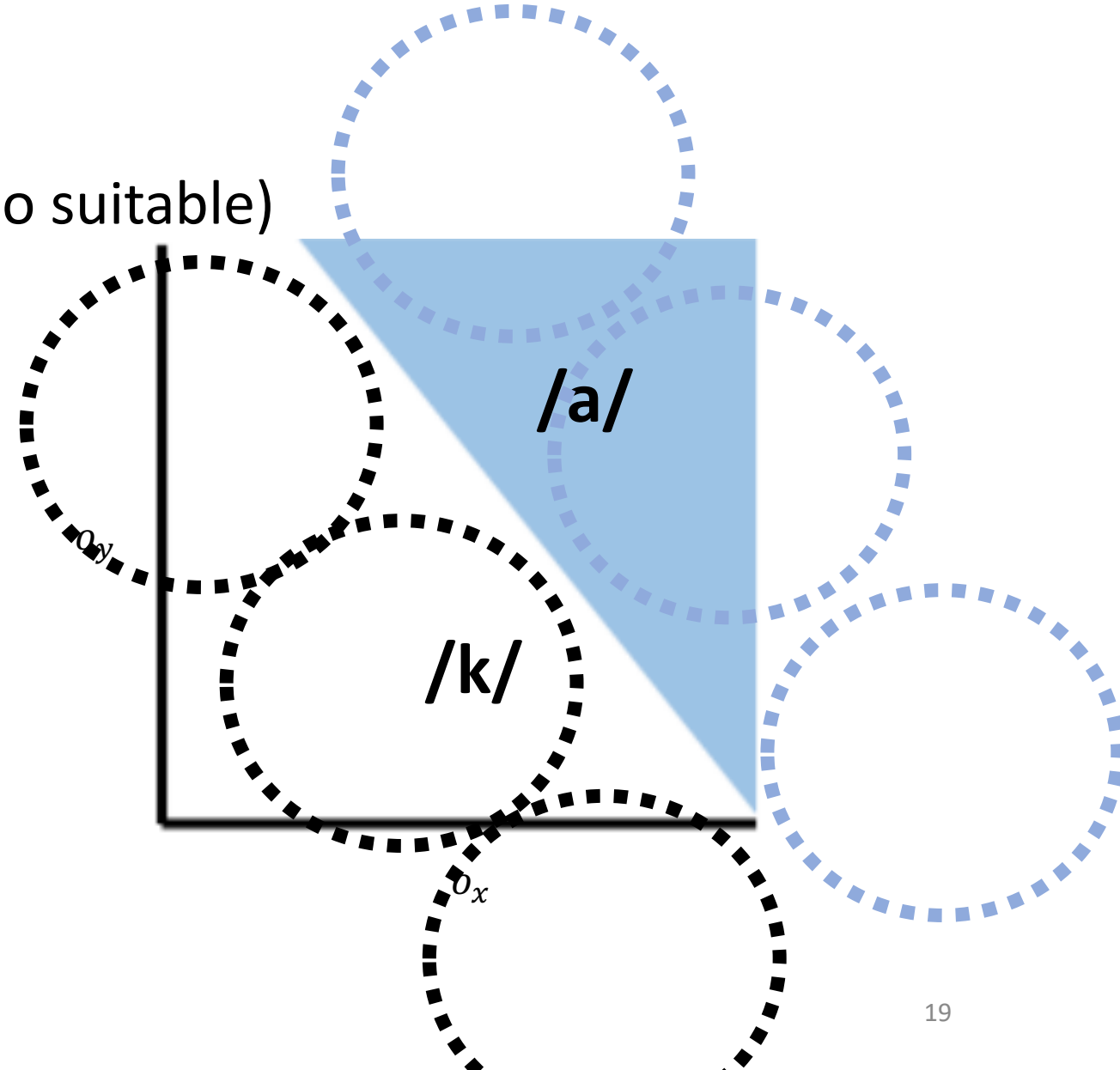
Very easy case

- We can use GMM (although not so suitable)

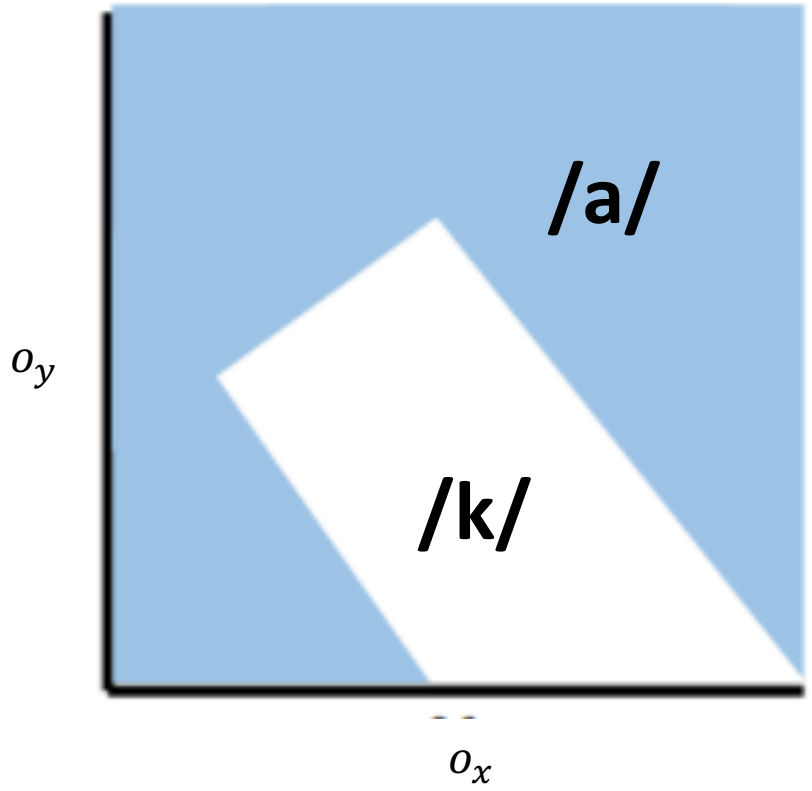
$$p(\mathbf{o}|/\text{a}/) = \sum_k \omega_k N(\mathbf{o}|\mu_k, \Sigma_k)$$

$$p(\mathbf{o}|/\text{k}/) = \sum_k \omega'_k N(\mathbf{o}|\mu'_k, \Sigma'_k)$$

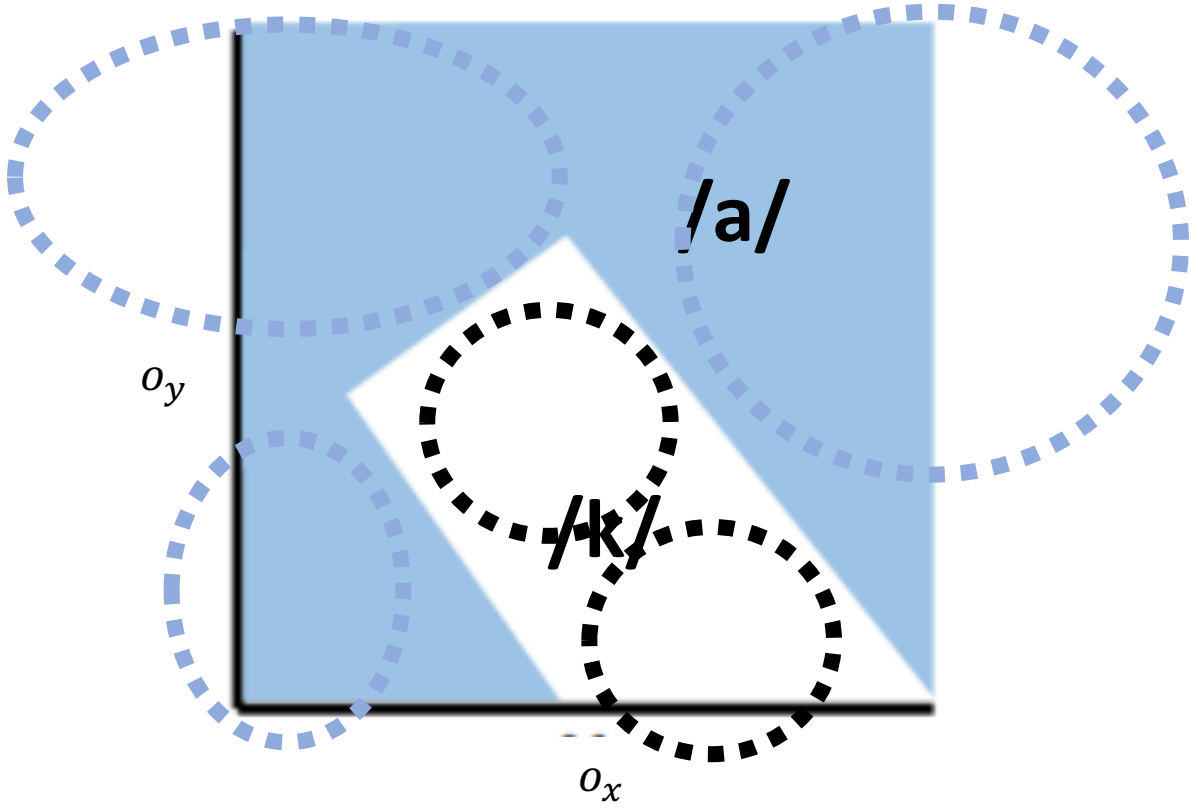
- $p(/\text{a}/|\mathbf{o}) \approx \frac{p(\mathbf{o}|/\text{a}/)}{p(\mathbf{o}|/\text{a}/)+p(\mathbf{o}|/\text{k}/)}$



Getting more difficult with the GMM classifier or linear classifier

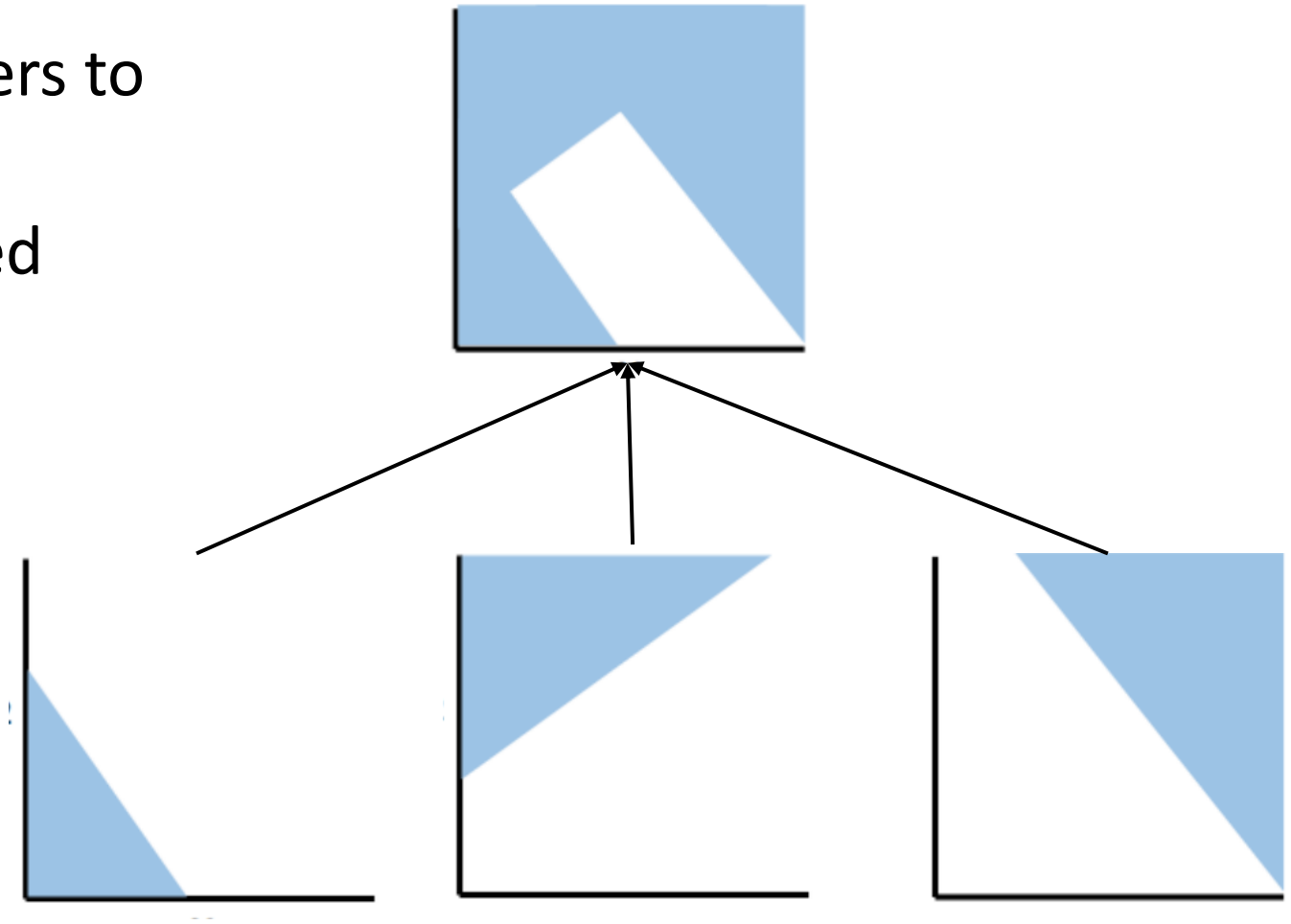


Getting more difficult with the GMM classifier or linear classifier



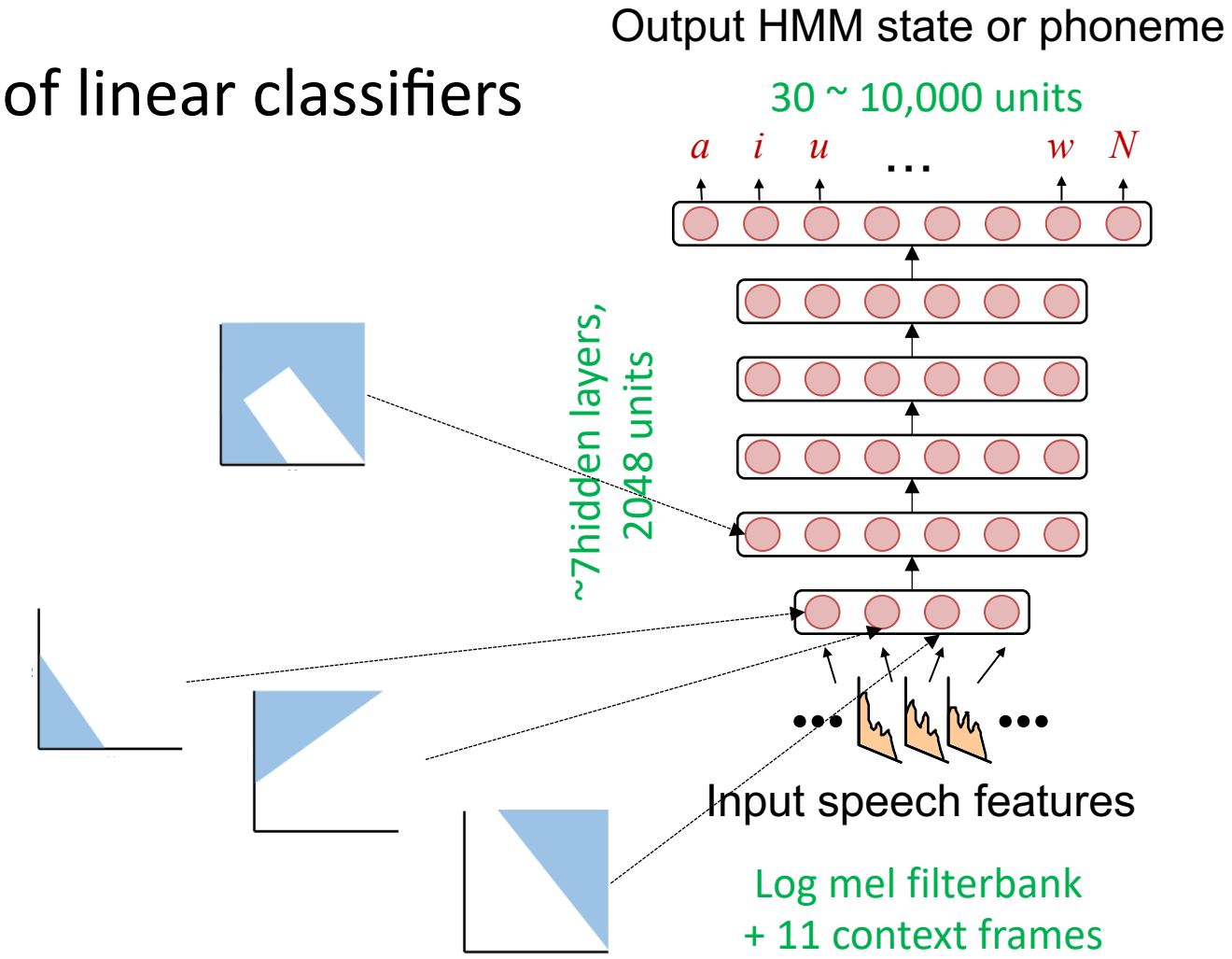
Neural network

- Combination of linear classifiers to classify complicated patterns
- More layers, more complicated patterns



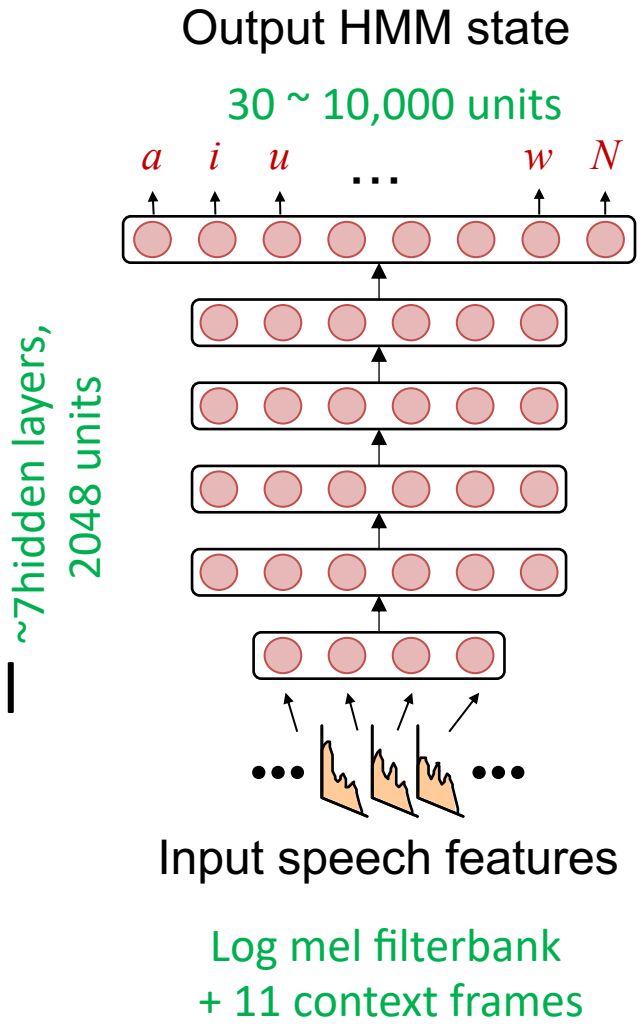
Neural network used in speech recognition

- Very large combination of linear classifiers



Why neural network was not focused

1. Very difficult to train
   - Batch? On-line? Mini-batch?
   - Stochastic gradient decent
     - Learning rate? Scheduling?
   - What kind of topologies?
   - Large computational cost
2. The amount of training data is very critical
3. CPU -> GPU



Before deep learning (2002 – 2009)

- Success of neural networks was very old period
- People believed that GMM was better
- But very small gain from standard GMMs

from https://en.wikipedia.org/wiki/Geoffrey_Hinton

When I noticed deep learning (2010)

- A. Mohamed, G. E. Dahl, and G. E. Hinton, “Deep belief networks for phone recognition,” in NIPS Workshop on Deep Learning for Speech Recognition and Related Applications, 2009.

Table 4: *Reported results on TIMIT core test set*

Method	PER
Stochastic Segmental Models [28]	36%
Conditional Random Field [29]	34.8%
Large-Margin GMM [30]	33%
CD-HMM [4]	27.3%
Augmented conditional Random Fields [4]	26.6%
Recurrent Neural Nets [31]	26.1%
Bayesian Triphone HMM [32]	25.6%
Monophone HTMs [33]	24.8%
Heterogeneous Classifiers [34]	24.40%
Deep Belief Networks(DBNs) (this work)	**23.0%**

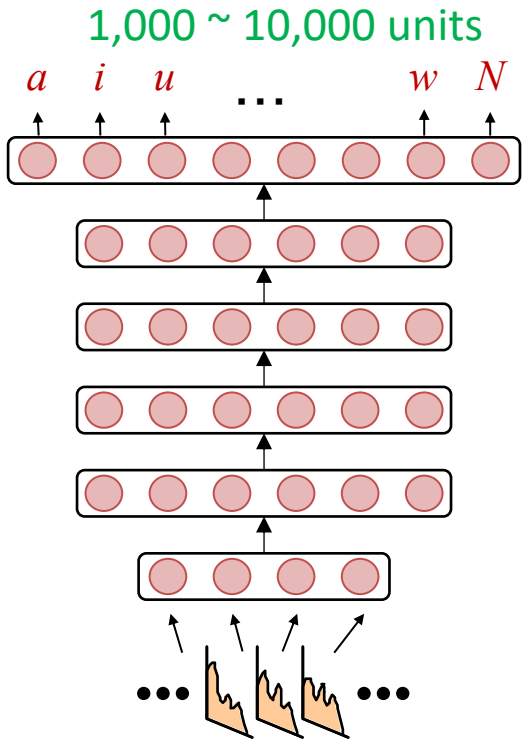
- Using deep belief network as **pre-training**
- **Fine-tuning** deep neural network

→ Provides stable estimation

- This still did not fully convince me (I introduced it at NTT’s reading group)

Pre-training and fine-tuning

- First train neural network like parameters with deep belief network or autoencoder
- Then, using deep neural network training



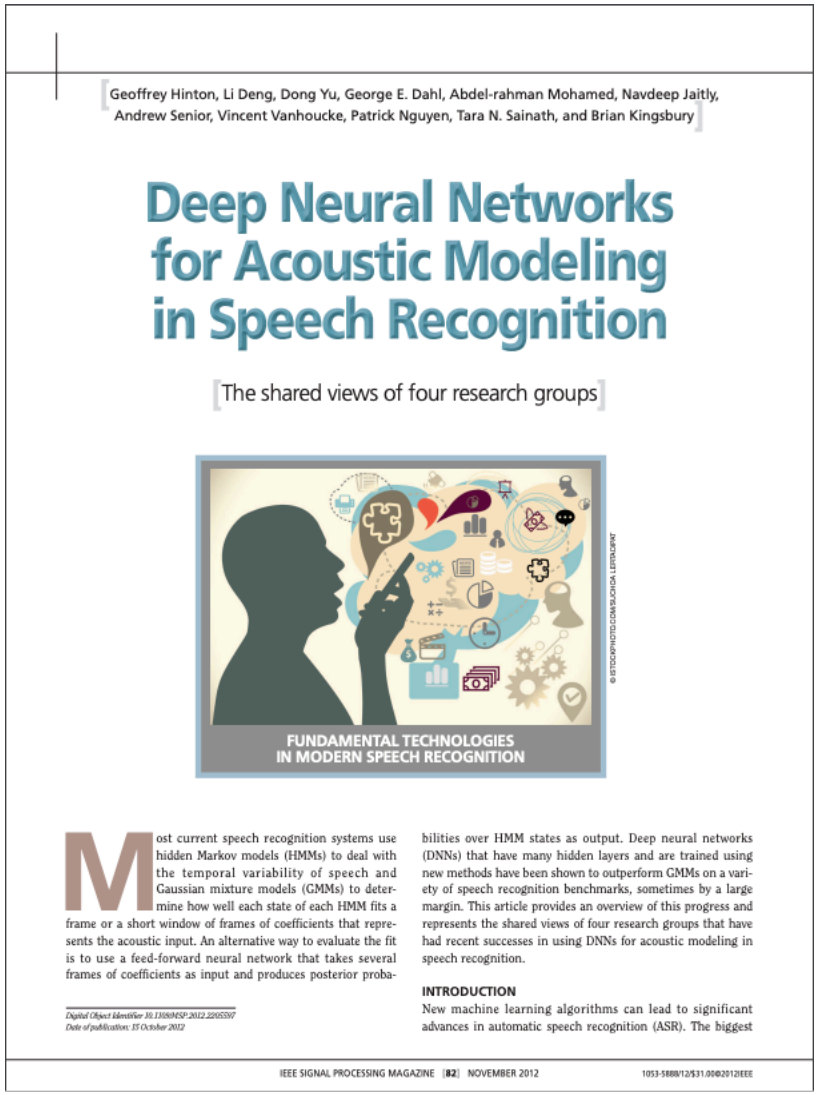
Interspeech 2011 at Florence

- The following three papers convinced me
  - Feature extraction: Valente, Fabio / Magimai-Doss, Mathew / Wang, Wen (2011): "Analysis and comparison of recent MLP features for LVCSR systems", In *INTERSPEECH-2011*, 1245-1248.
  - Acoustic model: Seide, Frank / Li, Gang / Yu, Dong (2011): "Conversational speech transcription using context-dependent deep neural networks", In INTERSPEECH-2011, 437-440.
  - Language model: Mikolov, Tomáš / Deoras, Anoop / Kombrink, Stefan / Burget, Lukáš / Černocký, Jan (2011): "Empirical evaluation and combination of advanced language modeling techniques", In INTERSPEECH-2011, 605-608.
- I discussed this potential to my NLP folks in NTT but they did not believe it (SVM, log linear model)

Late 2012

- My first deep learning (Kaldi nnet)
  - Kaldi started to support DNN since 2012 (mainly developed by Karel Vesely)
    - Deep belief network based pre-training
    - Feed forward neural network
    - Sequence-discriminative training

	Hub5 '00 (SWB)	WSJ
GMM	18.6	5.6
DNN	14.2	3.6
DNN with sequence-discriminative training	12.6	3.2



Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury

Deep Neural Networks for Acoustic Modeling in Speech Recognition

The shared views of four research groups

Most current speech recognition systems use hidden Markov models (HMMs) to deal with the temporal variability of speech and Gaussian mixture models (GMMs) to determine how well each state of each HMM fits a frame or a short window of frames of coefficients that represents the acoustic input. An alternative way to evaluate the fit is to use a feed-forward neural network that takes several frames of coefficients as input and produces posterior probabilities over HMM states as output. Deep neural networks (DNNs) that have many hidden layers and are trained using new methods have been shown to outperform GMMs on a variety of speech recognition benchmarks, sometimes by a large margin. This article provides an overview of this progress and represents the shared views of four research groups that have had recent successes in using DNNs for acoustic modeling in speech recognition.

Digital Object Identifier 10.1109/MSP.2012.2205597
Date of publication: 15 October 2012

INTRODUCTION

New machine learning algorithms can lead to significant advances in automatic speech recognition (ASR). The biggest

1053-5888/12/$31.00©2012IEEE

UNIVERSITY OF TORONTO

Microsoft

Google

IBM

Build speech recognition with public tools and resources

- TED-LIUM (~100 hours)
- LIBRISPEECH (~1000 hours)

- We can build use Kald+DNN+TED-LIUM to make English speech recognition system by using one machine (GPU + many core machines)
- Before this, it’s only for a big company.

Same things happened in *computer vision*

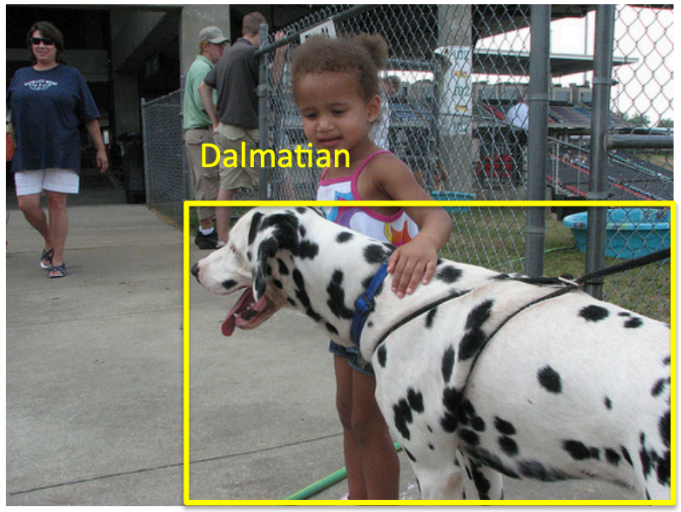
from https://en.wikipedia.org/wiki/Geoffrey_Hinton

ImageNet challenge (Large scale data)

IMAGENET Large Scale Visual Recognition Challenge (ILSVRC) 2010-2012

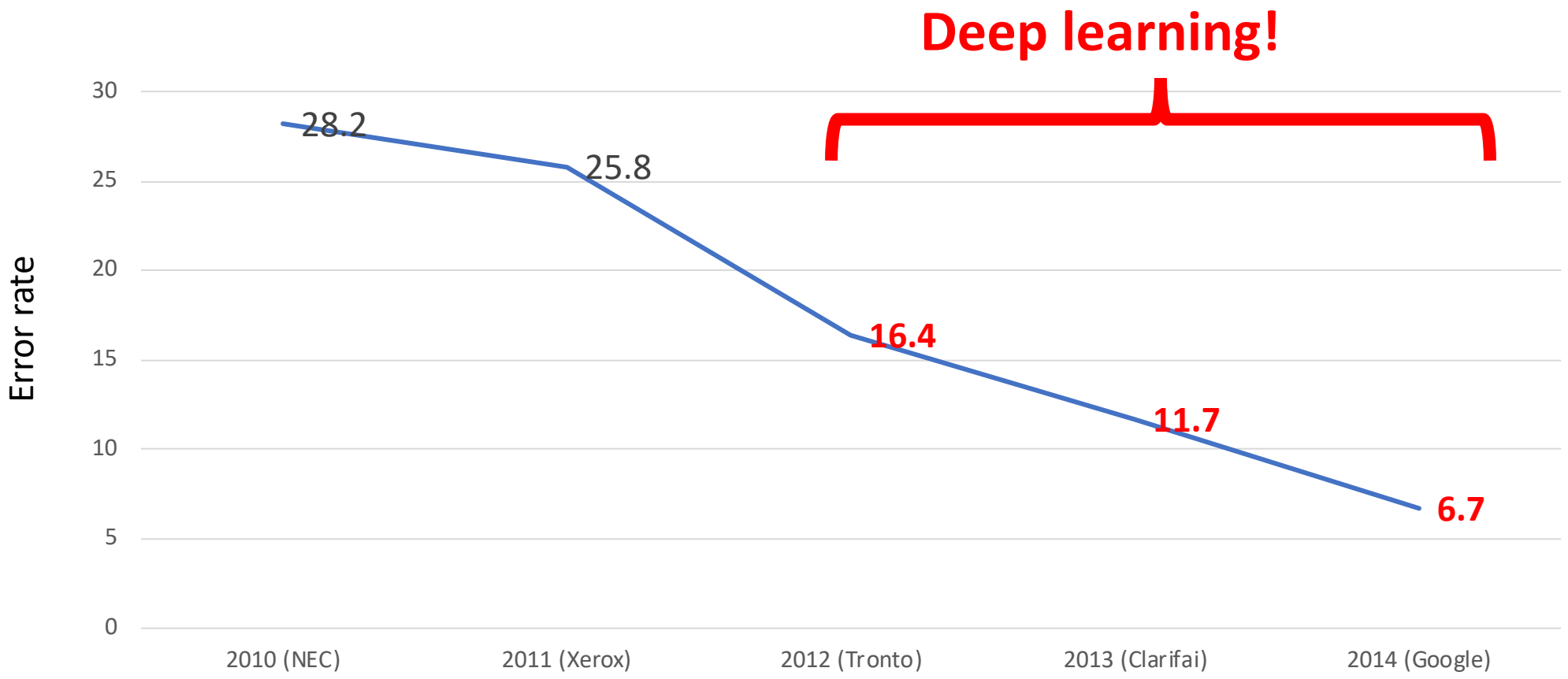
~~20 object classes~~ ~~22,591 images~~

1000 object classes **1,431,167 images**



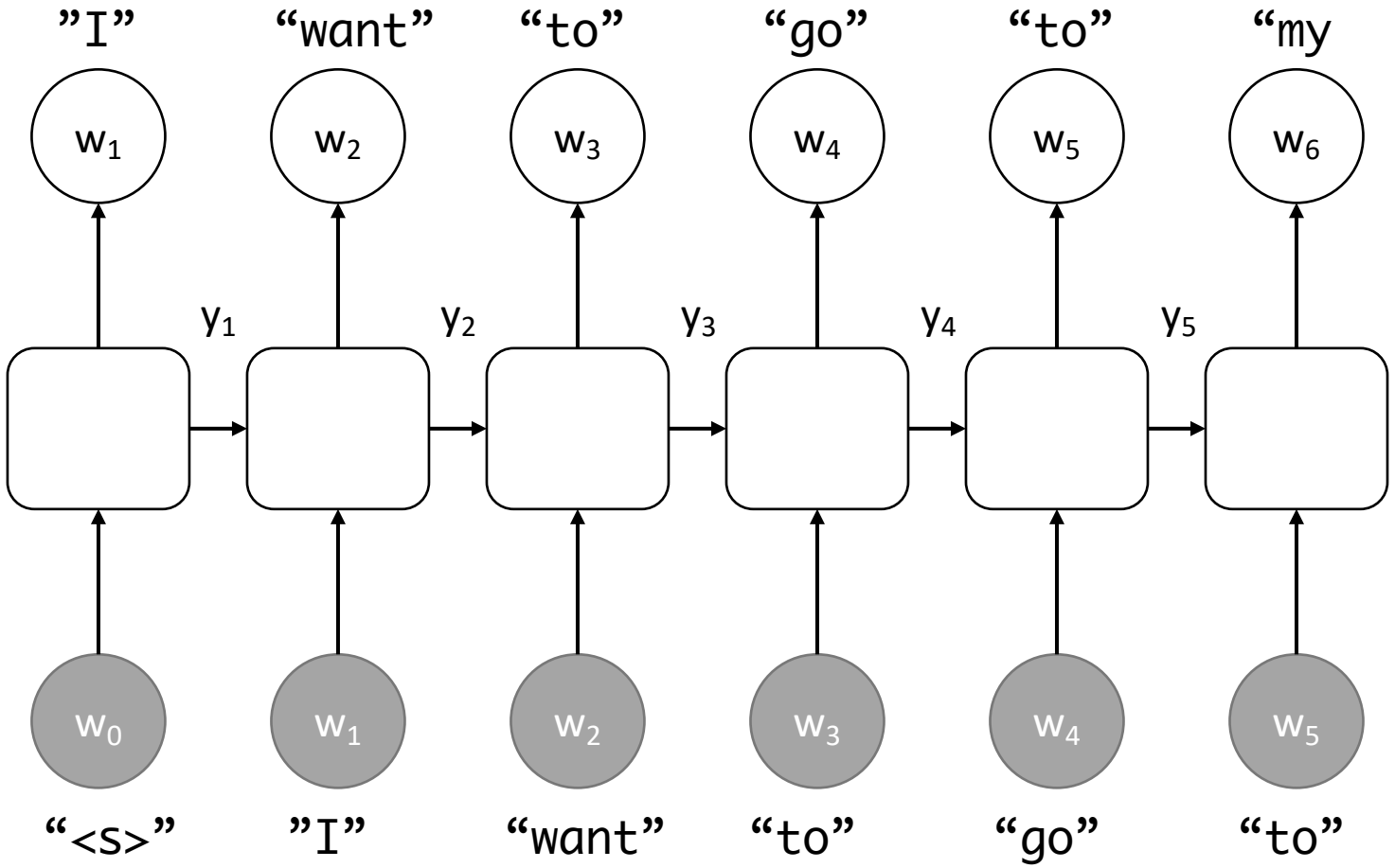
L. Fei-Fei and O. Russakovsky, **Analysis of Large-Scale Visual Recognition**, *Bay Area Vision Meeting, October, 2013*

ImageNet challenge
AlexNet, GoogLeNet, VGG, ResNet, …



Same things happened in *text processing*

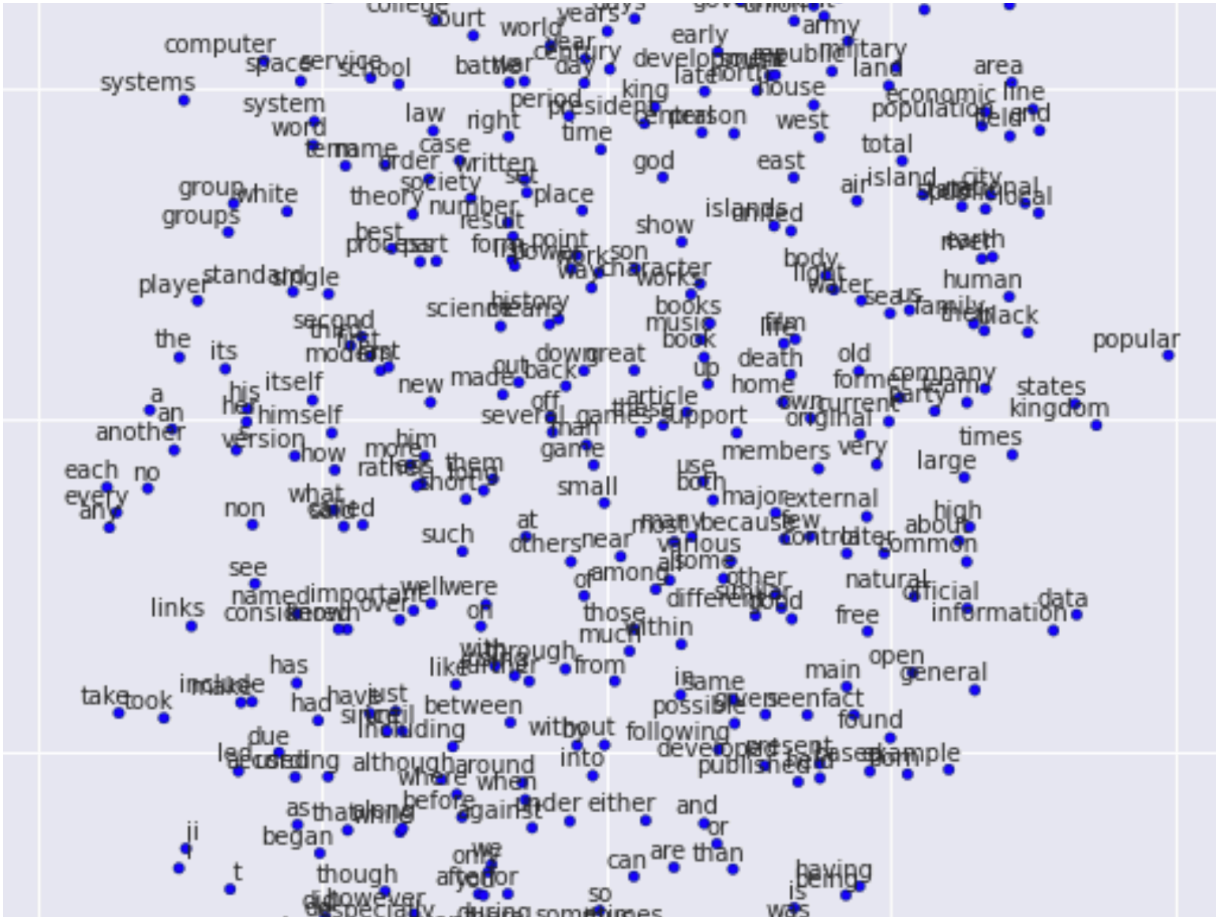
- Recurrent neural network language model (RNNLM) [Mikolov+ (2010)]



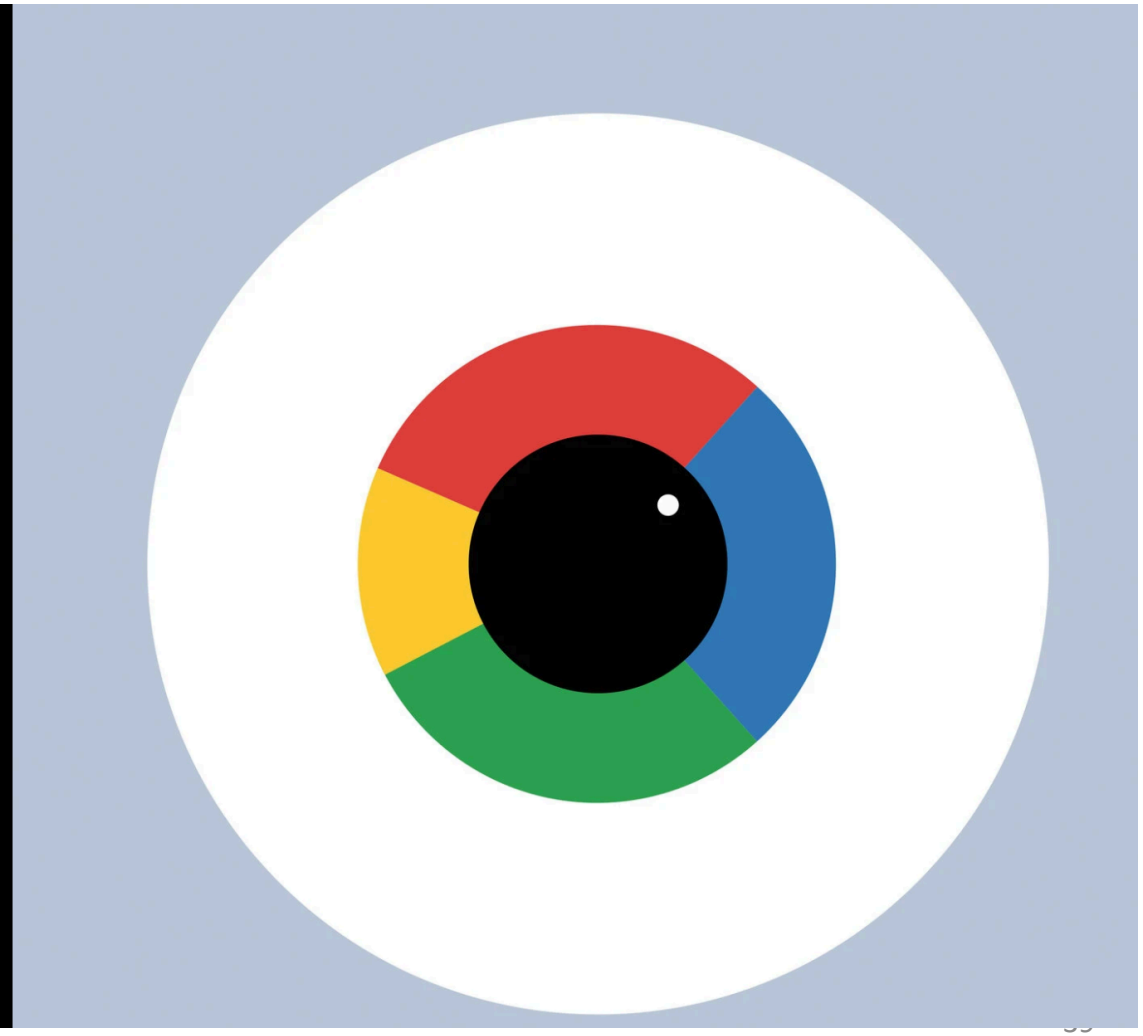
	Perplexity
N-gram (conventional)	336
RNNLM	**156**

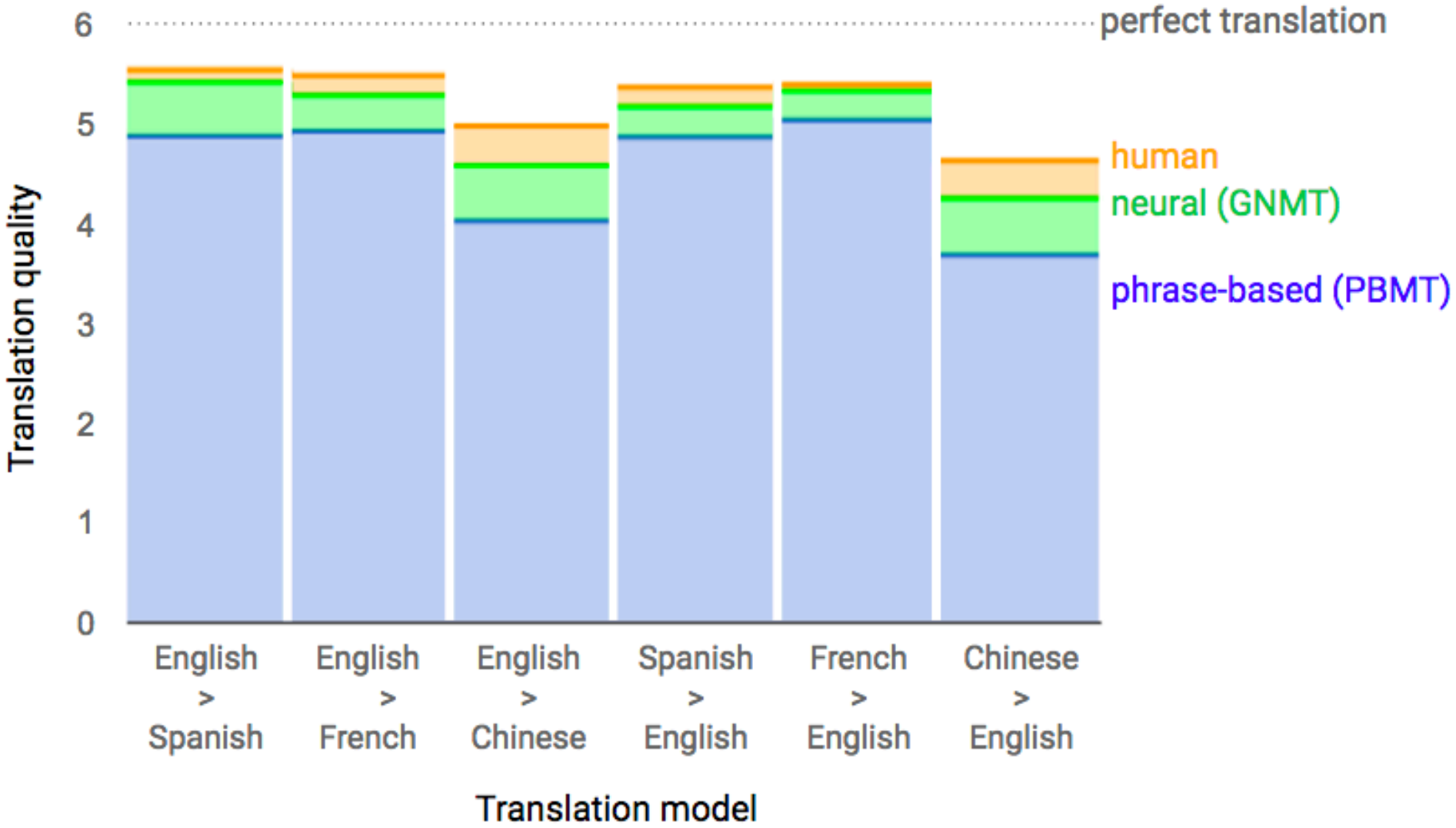
Word embedding example

https://www.tensorflow.org/tutorials/word2vec



Neural machine translation (New York Times, December 2016)





from https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

Deep neural network toolkit (2013-)

- Theano
- Caffe
- Torch
- CNTK
- Chainer
- Keras

Later

- Tensorflow
- **PyTorch (used in this course)**
- MxNet
- Etc.

Summary

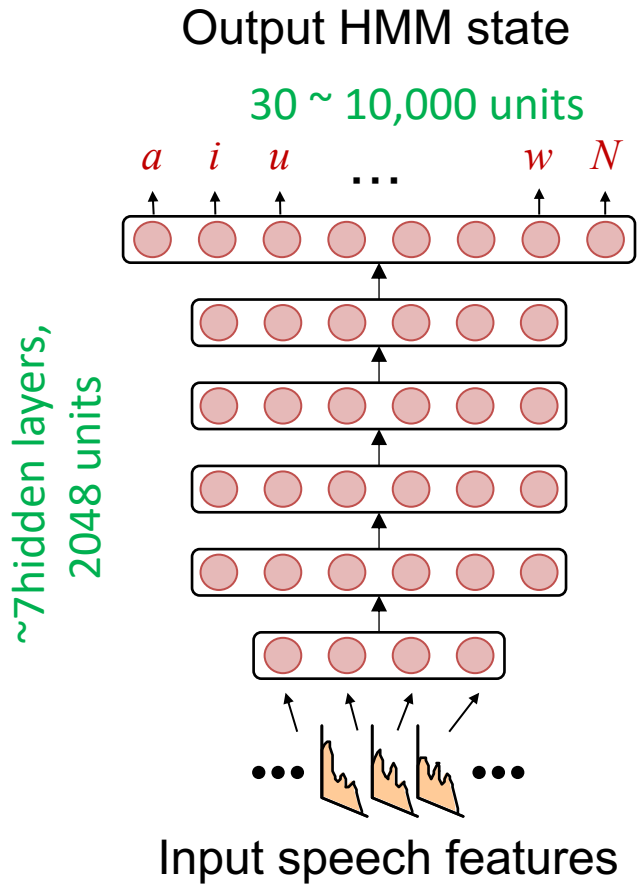
- Before 2011
  - GMM/HMM, limitation of the performance, bit boring
  - This is because they are linear models…
- After 2011
  - DNN/HMM
  - Toolkit
  - Public large data
  - GPU
  - NLP, image/vision were also moved to DNN
  - Always something exciting

Today’s agenda

- Introduction of deep neural network
- Basics of neural network

Feed-forward neural network for acoustic model

- Configurations
  - Input features
    - Context expansion
  - Output class
    - Softmax function
    - Training criterion
  - Number of layers
  - Number of hidden states
  - Type of non-linear activations



Input feature

- GMM/HMM formulation
  - Lot of conditional independence assumption and Markov assumption
  - Many of our trials are how to break these assumptions
- In GMM, we always have to care about the correlation
  - Delta, linear discriminant analysis, semi-tied covariance
- In DNN, we don't have to care ☺
  - We can simply concatenate the left and right contexts, and just throw it!

$$\mathbf{o}_t^{(2)} = \begin{bmatrix} \mathbf{o}_{t-r}^{(1)} \\ \vdots \\ \mathbf{o}_t^{(1)} \\ \vdots \\ \mathbf{o}_{t+r}^{(1)} \end{bmatrix}$$

Output

- Phoneme or HMM state ID is used
- We need to have a pair data of output and input data at frame t
  - First use the Viterbi alignment to obtain the state sequence

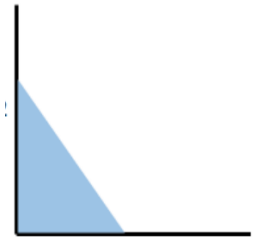
$$\hat{S} = \{\hat{s}_t | t = 1, \cdots, T\} = \underset{S}{\operatorname{argmax}}\, p(S, \mathbf{O} | \Theta_{\mathrm{gmm/hmm}})$$

  - Then, we get the input and output pair $\{\hat{s}_t, \mathbf{o}_t\}$ for all t
- Make acoustic model as a multiclass classification problem by predicting the all HMM state ID given the observation
  - Not consider any constraint in this stage (e.g., left to right, which is handled by an HMM during recognition)

Feed-forward neural networks

- Affine transformation and non-linear activation function (sigmoid function)

$$\mathbf{h}_t^{(1)} = \sigma(\mathbf{W}^{(1)}\mathbf{o}_t + \mathbf{b}^{(1)})$$



- Apply the above transformation L times

$$\mathbf{h}_t^{(l)} = \sigma(\mathbf{W}^{(l)}\mathbf{h}_t^{(l-1)} + \mathbf{b}^{(l)})$$

- Softmax operation to get the probability distribution

$$\{p(s_t = j|\mathbf{o}_t)\}_{j=1}^{J} = \text{softmax}(\mathbf{W}^{(L)}\mathbf{h}_t^{(L-1)} + \mathbf{b}^{(L)})$$

Linear operation

- Transforms $D^{(l-1)}$-dimensional input to $D^{(l)}$ output

$$f(\mathbf{h}^{(l-1)}) = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}$$

  - $\mathbf{W}^{(l)} \in \mathbb{R}^{D^{(l)} \times D^{(l-1)}}$: Linear transformation matrix
  - $\mathbf{b}^{(l)} \in \mathbb{R}^{D^{(l)}}$: bias vector
- Derivatives
  - $\frac{\partial \sum_j w_{ij} h_j + b_i}{\partial b_{i'}} = \delta(i, i')$
  - $\frac{\partial (\sum_j w_{ij} h_j + b_i)}{\partial w_{i'j'}} = \delta(i, i') h_{j'}$

Sigmoid function

- Sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

  - Convert the domain from $\mathbb{R}$ to $[0, 1]$
  - Elementwise sigmoid function:

$$\sigma(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}}} = \left[\frac{1}{1 + e^{-x_d}}\right]_{d=1}^{D}$$

  - No trainable parameter in general
- Derivative
  - $\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$

Softmax function

- Softmax function

$$p(j|\mathbf{h}) = [\text{softmax}(\mathbf{h})]_j = \frac{e^{h_j}}{\sum_{i=1}^{J} e^{h_i}}$$

  - Convert the domain from $\mathbb{R}^J$ to $[0,1]^J$ (make a multinomial dist. → classification)
  - Satisfy the sum to one condition, i.e., $\sum_{j=1} p(j|\mathbf{h}) = 1$
  - $J = 2$: sigmoid function
- Derivative
  - For $i = j$: $\frac{\partial p(j|\mathbf{h})}{\partial h_i} = p(j|\mathbf{h})(1 - p(j|\mathbf{h}))$
  - For $i \neq j$: $\frac{\partial p(j|\mathbf{h})}{\partial h_i} = -p(i|\mathbf{h})\, p(j|\mathbf{h})$
  - Or we can write as $\frac{\partial p(j|\mathbf{h})}{\partial h_i} = p(j|\mathbf{h})(\delta(i,j) - p(i|\mathbf{h}))$ $:\delta(i,j)$: Kronecker's delta

What functions/operations we cannot use?

- The function/operations that we cannot take a derivative, including some discrete operation
  - $\text{argmax}_w p(W|O)$: Basic ASR operation, but we cannot take a derivative….
  - Discretization
  - Etc.

Objective function design

- We usually use the cross entropy as an objective function

$$
\begin{aligned}
\mathcal{C}_{\mathrm{CE}}(\Theta_{\mathrm{dnn}}) &= \sum_t \mathrm{CE}[p^{\mathrm{ref}}(s_t)|p(s_t|\mathbf{o}_t, \Theta_{\mathrm{dnn}})] \\
&= -\sum_t \sum_{s_t} p^{\mathrm{ref}}(s_t) \log p(s_t|\mathbf{o}_t, \Theta_{\mathrm{dnn}})] \\
&= -\sum_t \sum_{s_t} \delta(s_t, \hat{s}_t) \log p(s_t|\mathbf{o}_t, \Theta_{\mathrm{dnn}})] \\
&= -\sum_t \log p(\hat{s}_t|\mathbf{o}_t, \Theta_{\mathrm{dnn}})]
\end{aligned}
$$

  - Since the Viterbi sequence is a hard assignment, the summation over states is simplified

Other objective functions

- Square error

$$\left|\mathbf{h}^{\text{ref}} - \mathbf{h}\right|^2$$

  - We could also use p norm, e.g., L1 norm
- Binary cross entropy

$$\begin{aligned}\mathcal{C}_{\text{Binary}}(\Theta_{\text{dnn}}) &= -\sum_t \log p(\hat{s}_t|\mathbf{o}_t, \Theta_{\text{dnn}})] \\ &= -\sum_t \log \sigma(\mathbf{h}_t)\end{aligned}$$

  - Again this is a special case of the cross entropy when the number of classes is two

Building blocks

Output: $s_t \in \{1, \ldots, J\}$

Softmax activation

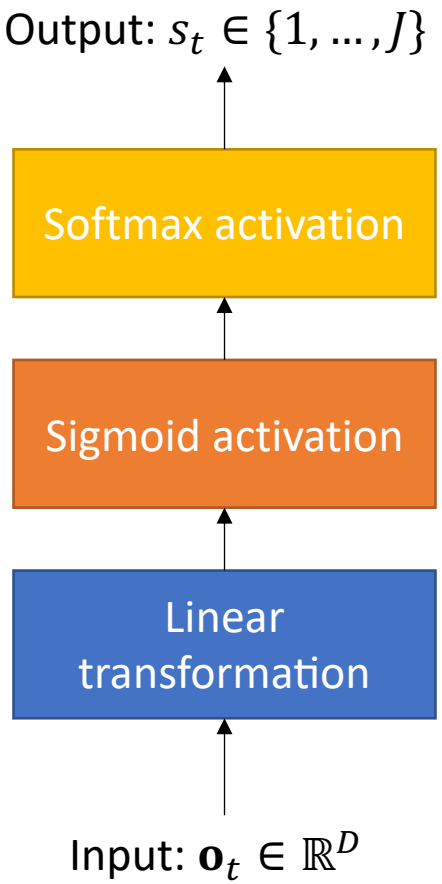
Sigmoid activation

Linear transformation

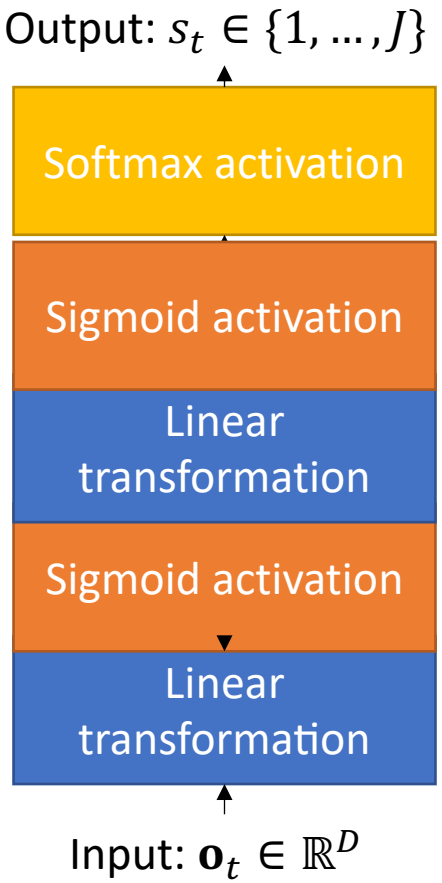
$+, -, \exp(\quad), \log(\quad)$, etc.

Input: $\mathbf{o}_t \in \mathbb{R}^D$

Building blocks



Building blocks



Building blocks

Output: $s_t \in \{1, \ldots, J\}$

Softmax activation

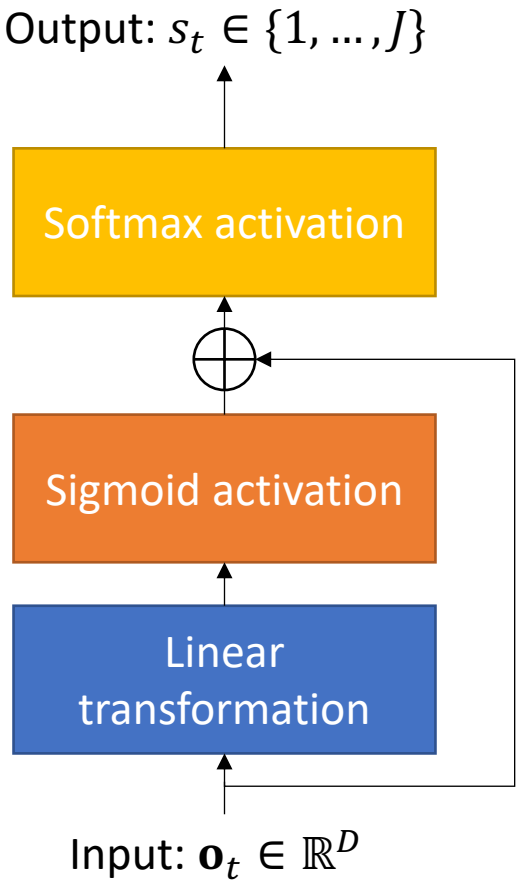
Sigmoid activation

Linear transformation

$+, -, \exp(\quad), \log(\quad)$, etc.

Input: $\mathbf{o}_t \in \mathbb{R}^D$

Building blocks



How to optimize?
Gradient decent and their variants

- Take a derivative and update parameters with this derivative

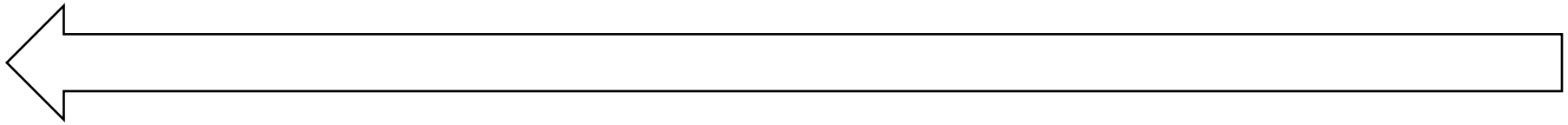
$$\Theta_{\text{dnn}}^{(\text{new})} = \Theta_{\text{dnn}}^{(\text{old})} - \rho \frac{\partial}{\partial \Theta_{\text{dnn}}} \mathcal{C}_{\text{CE}}(\Theta_{\text{dnn}}) \Big|_{\Theta_{\text{dnn}}=\Theta_{\text{dnn}}^{(\text{old})}}$$

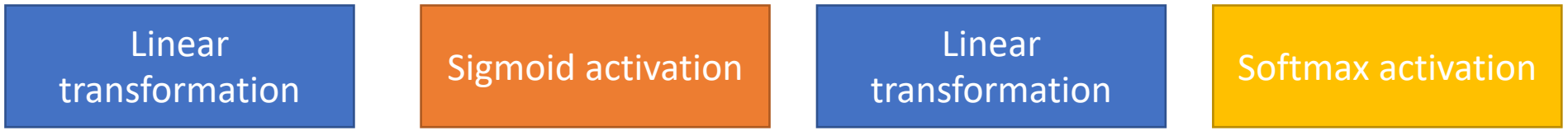
- Chain rule

$$\frac{\partial}{\partial \theta} f(g(\theta)) = \frac{\partial}{\partial g} \frac{\partial g}{\partial \theta} f(g(\theta)) = f'(g(\theta)) g'(\theta)$$

Deep neural network: nested function

- Chain rule to get a derivative recursively
  - Each transformation (Affine, sigmoid, and softmax) has analytical derivatives and we just combine these derivatives
  - We can obtain the derivative from the back propagation algorithm





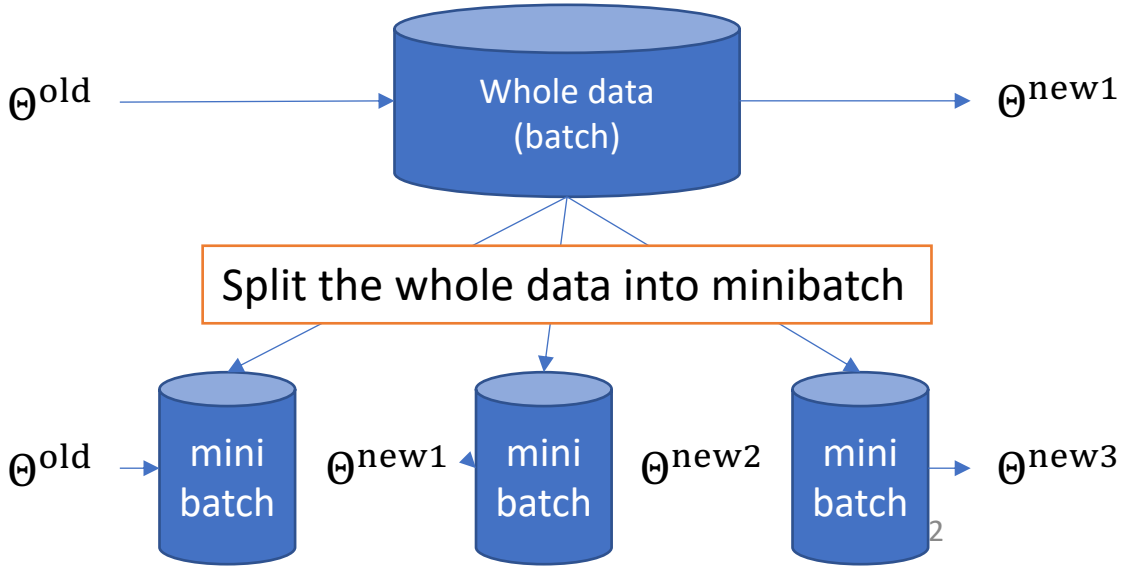
Minibatch processing

- Batch processing
  - Slow convergence
  - Effective computation
- Online processing
  - Fast convergence
  - Very inefficient computation
- Minibatch processing
  - Something between batch and online processing

$$\Theta_{\text{dnn}}^{(\text{new})} = \Theta_{\text{dnn}}^{(\text{old})} - \rho \frac{\partial}{\partial \Theta_{\text{dnn}}} \mathcal{C}_{\text{CE}}(\Theta_{\text{dnn}}) \Big|_{\Theta_{\text{dnn}}=\Theta_{\text{dnn}}^{(\text{old})}}$$

where

$$\mathcal{C}_{\text{CE}}(\Theta_{\text{dnn}}) = -\sum_{t} \log p(\hat{s}_t | \mathbf{o}_t, \Theta_{\text{dnn}})$$



Summary of today's talk

- Deep learning changes the world
- A lot of human language technologies are boosted by deep learning
- Deep neural network basics
  - Input
  - Output
  - Function
  - Back propagataion